2008

Efficient Mining of Heterogeneous Star-Structured Data

Manjeet Rege

Qi Yu

Recommended Citation
Int J Software Informatics, Vol.2, No.2, December 2008, pp. 141{161

Int J Software Informatics, Vol.2, No.2, December 2008, pp. 141–161 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
 Tel: +86-10-62661040

Efficient Mining of Heterogeneous Star-Structured Data*

Manjeet Rege[1] and Qi Yu[2]

[1](Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA)

[2](Department of Information Technology, Rochester Institute of Technology, Rochester, NY, USA)

Abstract Many of the real world clustering problems arising in data mining applications are heterogeneous in nature. Heterogeneous co-clustering involves simultaneous clustering of objects of two or more data types. While pairwise co-clustering of two data types has been well studied in the literature, research on high-order heterogeneous co-clustering is still limited. In this paper, we propose a graph theoretical framework for addressing star-structured co-clustering problems in which a central data type is connected to all the other data types. Partitioning this graph leads to co-clustering of all the data types under the constraints of the star-structure. Although, graph partitioning approach has been adopted before to address star-structured heterogeneous complex problems, the main contribution of this work lies in an efficient algorithm that we propose for partitioning the star-structured graph. Computationally, our algorithm is very quick as it requires a simple solution to a sparse system of overdetermined linear equations. Theoretical analysis and extensive experiments performed on toy and real datasets demonstrate the quality, efficiency and stability of the proposed algorithm.

Key words: Co-clustering; High-Order Heterogeneous Data; Isoperimetric; Consistency

Rege M, Yu Q. Efficient mining of heterogeneous star-structured data. *Int J Software Informatics*, **2008, 2(2): 141–161. http://www.ijsi.org/1673-7288/2/141.pdf**

1 Introduction

Clustering in general refers to organizing data objects into different groups (clusters) such that objects in one group are similar together and dissimilar from those in other groups under a predefined criteria of similarity. Homogeneous clustering consists of objects of a single data type and has been well studied in the literature for many years[1].

However, many of the real world clustering problems arising in data mining applications are heterogeneous in nature. Clustering problems of these kind have objects of more than one data type that need to be clustered together. A specific case of this is that of pairwise co-clustering of objects of two data types, such as documents and words in text corpus, patient symptoms and medical diagnosis in biomedical applications, customers and items purchased in customer relationship management

* Corresponding author: Manjeet Rege, Email: mr@cs.rit.edu
Manuscript received 10 Oct., 2008; revised 4 Dec., 2008; accepted 20 Dec., 2008; published online 29 Dec., 2008.

and business intelligence applications. Typically, the data is stored in a contingency or co-occurrence matrix where rows and columns of the matrix represent the data types to be co-clustered. An entry of this matrix signifies the relation between the data type represented by the corresponding row and the column. In Ref.[2], co-clustering is defined by a pair of maps from rows to row-clusters and from columns to column-clusters inducing clustered random variables. Optimal co-clustering is then derived based on the one that leads to the largest mutual information between the clustered random variables. Cai *et al.*[3] have applied this algorithm to co-cluster auditory scenes and audio elements for unsupervised content discovery in audio. The minimum Bregman information principle is proposed in Ref.[4] as a generalization of the maximum entropy principle. Based on this principle, an algorithm for the Bregman co-clustering problem is developed. George and Merugu[5] adapted the Bregman co-clustering algorithm to a collaborative filtering framework. Apart from these, there have been works on graph partitioning based pairwise co-clustering such as the popular spectral co-clustering[6,7] or the recent isoperimetric co-clustering[8].

While pairwise co-clustering has been well studied as seen from the above discussion, research on high-order heterogeneous co-clustering is limited. Zeng *et al.*[9] proposed a framework for clustering heterogeneous Web objects, where a layered structure with link information is used to iteratively project and propagate the cluster results between layers. Similarly, in Ref.[10] an approach to improve the cluster quality of interrelated data objects through an iterative reinforcement clustering process is presented. However, there is no sound objective function and theoretical proof on effectiveness and convergence of these algorithms. Recently, Long *et al.*[11,12] formulated heterogeneous co-clustering as collective factorization on related matrices. Iterative algorithms presented in these works cluster each of the data types in different number of clusters. Consequently, they do not provide a one-one correspondence information between the data types from co-clustering results.

In this paper, we propose the Consistent Isoperimetric High-Order Co-clustering (CIHC) framework for addressing star-structured co-clustering problems in which a central data type is connected to all the other data types as shown in Fig.1(a). Intuitively, one might expect that this kind of co-clustering can be achieved by trivially extending the pairwise co-clustering techniques previously developed. However, this approach does not consider the constraints enforced on the central data type. A basic element of this problem is the star-structured triplet of Fig.1(b). It is easy to see that, a framework for co-clustering the triplet can be extended to high-order star-structured problems. We model the triplet using a tripartite graph consisting of the 3 data types - X, Y and Z. We treat this tripartite graph as two bipartite graphs of X-Y and Y-Z. Co-clustering is then achieved by simultaneously partitioning these two bipartite graphs together such that the star-structured constraint is respected. Note that the simultaneous partitioning of the two bipartite graphs is performed in such a way that the local clustering of each graph need not be optimal under the constraint that the fusion of the two results yields optimum co-clustering of X, Y and Z. Actually, a similar concept was presented by Gao *et al.*[13,14] where the Consistent Bipartite Graph Co-partitioning (CBGC) was proposed under the spectral graph partitioning paradigm. An iterative algorithm using semi-definite programming (SDP)[15] was used to partition the tripartite graph which is computationally expensive and does not work well

on large data sets. On the other hand, the proposed methodology requires a simple solution to a sparse system of overdetermined linear equations. Moreover, the CIHC framework has been derived from isoperimetric graph partitioning which has been shown to achieve superior results than the spectral approach[8,16,17]. Experimental results performed on toy and real datasets demonstrate the advantages of CIHC over CBGC in co-clustering star-structured problems.

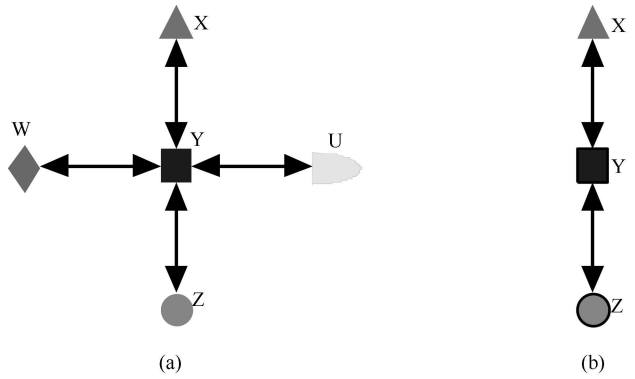

Figure 1. (a) Star-structured co-clustering problem where a central data type is connected to other data types. (b) Star-structured triplet is a basic element of this co-clustering problem

Rest of the paper is organized as follows. Section 2 provides an overview on graph partitioning and reviews the related work in the literature. The proposed CIHC framework is presented in Section 3. Theoretical analysis of the proposed framework is performed in Section 4. Experimental results are presented in Section 5. Finally, conclusions and future work directions appear in Section 6.

2 Related Work

In this section, we introduce some essential background on graph theory and review related work in the literature.

2.1 Homogeneous graphs for clustering

An undirected homogeneous graph G ={V, E} consists of a set of vertices $V = \{v_1, v_2, ..., v_{|V|}\}$ and a set of edges E=$\{e_{ij}|$ edge between v_i and v_j, $i, j <= |V|\}$, where $|V|$ is the number of vertices. In a weighted graph, each edge e_{ij} has a positive weight denoted by $w(e_{ij})$. The weight of the edge signifies the level of association between the vertices. An edge weight of zero denotes the absence of an edge between the two respective vertices. Given a vertex numbering and the edge weights between the vertices, graphs can be represented by matrices. The adjacency matrix **J** of the graph is defined as,

$$J_{ij} = \begin{cases} w(e_{ij}), & \text{if } e_{ij} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The degree of a vertex v_i denoted by d_i is defined as,

$$d_i = \sum_{e_{ij}} w(e_{ij}), \qquad \forall e_{ij} \in E \tag{2.2}$$

The degree matrix **D** of the graph is a diagonal matrix having degree of vertices along the diagonal while a degree vector **d** of a graph is a vector consisting of degree of all the vertices. The Laplacian matrix **L** of a graph is a symmetric matrix with one row and column for each vertex such that,

$$L_{v_i,v_j} = \begin{cases} d_i, & \text{if } i = j \\ -w(e_{ij}), & \text{if } e_{ij} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

A graph partitioning algorithm assigns a set of values to each vertex in the graph. We will refer to a vector consisting of the values for each of the vertices as the *indicator vector* of the graph. The *cutting* of the graph is dividing the indicator vector based on the values associated with each vertex using a splitting value. If **u** denotes the indicator vector of the graph and s is the splitting value, then the vertices are partitioned into the set of i such that $u_i > s$ and the set such that $u_i \leqslant s$. Spectral graph theory[18] which is based on performing eigen decomposition on matrices of the graphs, has been one of the most popular and widely applied graph partitioning methods. In[19], spectral graph partitioning was applied to image segmentation by solving the generalized eigenvalue problem[18,20],

$$\mathbf{Lx} = \lambda \mathbf{D}\ \mathbf{x} \tag{2.4}$$

Partitions are then obtained by running a clustering algorithm such as k-means[1] on the eigenvector **x** corresponding to second smallest eigenvalue λ_2.

2.2 Bipartite graph model for pairwise co-clustering

An undirected bipartite graph $G = \{Y, Z, E\}$, has two sets of vertices, viz., Y and Z and a set of graph edges E. Let **B** be an $|Y|$ by $|Z|$ graph weight matrix. An entry B_{ij} in this matrix is the weight of an edge appearing between a vertex $y_i \in Y$ and a vertex $z_j \in Z$. There are no edges between vertices of the same group. Then, the adjacency matrix of the bipartite graph is expressed as,

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \tag{2.5}$$

where the first $|Y|$ vertices index Y and the last $|Z|$ index Z. The two data types in the co-clustering problem can be represented by the two types of vertices of the weighted bipartite graph. Co-clustering of the data is achieved by partitioning the bipartite graph. In Fig.2, we show the bipartite graph partitioned using a dotted line. The two partitions obtained are $\{y_1, y_2, y_3, z_1, z_2, z_3\}$ and $\{y_4, y_5, y_6, z_4, z_5, z_6, z_7\}$, respectively. Therefore, the objects in Y are clustered into $\{y_1, y_2, y_3\}$ and $\{y_4, y_5, y_6\}$, while those in Z are clustered into $\{z_1, z_2, z_3\}$ and $\{z_4, z_5, z_6, z_7\}$ simultaneously. In order to compute these partitions using the spectral approach, we also need to solve a generalized eigenvalue problem as in equation (2.4). However, due to the bipartite

nature of the problem, the eigenvalue problem reduces to a much efficient singular value decomposition (SVD)[20]. This algorithm has found application in a wide range of co-clustering problems arising in fields such as text mining[6,7], Web search[21] and bioinformatics[22].

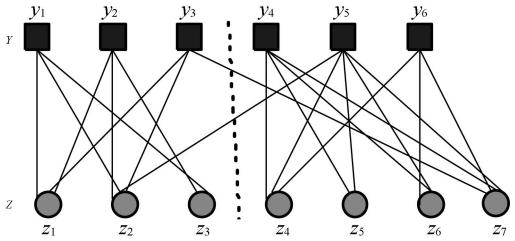

Figure 2. The square and circular vertices denote the two data types (Y and Z) in the pairwise co-clustering problem that are represented by the bipartite graph. Partitioning this bipartite graph leads to co-clustering of the two data types

Recently, Isoperimetric Co-clustering Algorithm (ICA)[8] was proposed to achieve pairwise co-clustering by partitioning a bipartite graph. ICA bears resemblance to the spectral approach in the sense that it does not require the coordinate information of the vertices of the graphs and allows us to find partitions of an optimal cardinality instead of a predefined cardinality. It has been shown that ICA outperforms the spectral approach in terms of the quality, efficiency and stability in partitioning a bipartite graph.

2.3 Tripartite graph model for triplet co-clustering

We model the star-structured triplet using a tripartite graph shown in Fig.3. An undirected tripartite graph $G = \{X, Y, Z, E\}$, has three sets of vertices, viz., X, Y and Z with E as the set of edges. If **A** and **B** represent the weight matrices for X-Y and Y-Z bipartite graphs respectively, then the adjacency matrix of the graph is defined as,

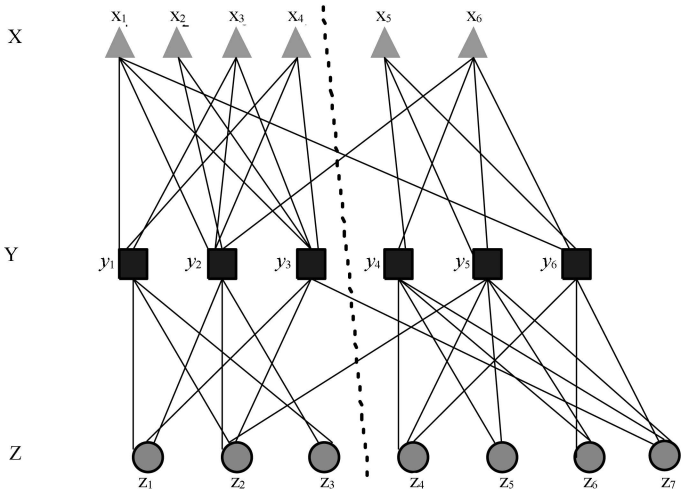

Figure 3. Tripartite graph of the three data types X, Y and Z of the star-structured triplet co-clustering

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{B}^T & \mathbf{0} \end{bmatrix} \tag{2.6}$$

Gao *et al.*[13,14] have used a similar tripartite graph model in their Consistent Bipartite Graph Co-partitioning (CBGC) framework. The tripartite graph is considered to be a fusion of the two bipartite graphs that are partitioned simultaneously using the spectral approach. Let $\mathbf{q} = [\mathbf{x} \quad \mathbf{y}]^T$ and $\mathbf{p} = [\mathbf{y} \quad \mathbf{z}]^T$ denote the indicator vectors for the two bipartite graphs and $\mathbf{D}^{(xy)}$, $\mathbf{D}^{(yz)}$, $\mathbf{L}^{(xy)}$ and $\mathbf{L}^{(yz)}$ represent the degree and Laplacian matrices of the two bipartite graphs. Then the objective function to minimize for the tripartite graph is expressed as a linear combination of objective functions of the two bipartite graphs as follows,

$$\begin{gathered} \min \left\{ \beta \frac{\mathbf{q}^T \mathbf{L}^{(xy)} \mathbf{q}}{\mathbf{q}^T \mathbf{D}^{(xy)} \mathbf{q}} + (1-\beta) \frac{\mathbf{p}^T \mathbf{L}^{(yz)} \mathbf{p}}{\mathbf{p}^T \mathbf{D}^{(yz)} \mathbf{p}} \right\} \\ \text{subject to } \mathbf{q}^T \mathbf{D}^{(xy)} \mathbf{e} = 0, \mathbf{q} \neq \mathbf{0} \\ \mathbf{p}^T \mathbf{D}^{(yz)} \mathbf{e} = 0, \mathbf{p} \neq \mathbf{0} \\ 0 < \beta < 1 \end{gathered} \tag{2.7}$$

where the parameter β specifies the weightage for each bipartite graph in the linear combination. Illumined by this work and the recent results of ICA for pair-wise co-clustering[8], we propose to partition the star-structured tripartite graph using isoperimetric graph partitioning.

3 Isoperimetric Graph Partitioning for Star-Structured Triplet

We partition the tripartite graph for star-structured triplet co-clustering by extending the ICA framework. To proceed, we first provide a brief overview of ICA to partition a bipartite graph.

ICA has been motivated from the combinatorial formulation of the classic isoperimetric problem[16,17,23−25]: *For a fixed area, find the shape with minimum perimeter.* It provides polynomial time heuristic for the NP-hard problem of finding a region with minimum perimeter for a fixed area. Let $V = \{Y \bigcup Z\}$ be the set of vertices of the bipartite graph. ICA partitions V into sets S and S^c, such that $S \bigcup S^c = V$ and $S \bigcap S^c = \phi$. Like other graph partitioning algorithms, ICA achieves optimum partitioning by finding S and S^c so that isoperimetric ratio of the graph h_G defined as,

$$h_G = \frac{|\Delta S|}{Vol_S} \tag{3.8}$$

is minimized. The numerator and denominator represent the boundary area and the volume of S, respectively. The boundary of S is defined as,
$\Delta S = \{e_{ij} |$ edges between a vertex in S and $S^c\}$. Consequently,

$$|\triangle S| = \sum_{e_{ij} \in \triangle S} w(e_{ij}) \tag{3.9}$$

The combinatorial volume[23,24] for the weighted bipartite graph is defined as:

$$Vol_S = |S|, \tag{3.10}$$

or,

$$Vol_S = \sum_i d_i, \forall \ vertices \in S, \tag{3.11}$$

where d_i is the degree of the vertex. The definition in equation (3.10) is known as the unit volume for a graph with uniform weights and the one in equation (3.11) is the degree volume for regular weighted graphs. After a few mathematical deductions, ICA achieves the minimization of equation (3.8) by solving a sparse system of linear equations instead of the eigenvalue or the singular value decomposition problem in the spectral approach. Depending on the notion of volume used for the partition, we get a system of linear equations. To partition a bipartite graph of Fig.2 we get,

$$\mathbf{L} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \mathbf{e}, \text{ for volume as per equation (3.10)} \tag{3.12}$$

or

$$\mathbf{L} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \mathbf{d}, \text{ for volume as per equation (3.11)} \tag{3.13}$$

where $\mathbf{L}$ is the Laplacian matrix of the bipartite graph, $[\mathbf{y}\ \mathbf{z}]^T$ the indicator vector to indicate partitioning of the two types of vertices, $\mathbf{d}$ the degree vector of the bipartite graph, and $\mathbf{e}$ a unit vector of length $|Y| + |Z|$. For the rest of the paper, we will represent both the notions together as,

$$\mathbf{L} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \mathbf{t} \tag{3.14}$$

Solving this system of equations results in a real valued $[\mathbf{y}\ \mathbf{z}]^T$. In order to get partitions, this solution needs to be *cut* using a *splitting value* (as explained in Section 2.

To partition the star-structured tripartite graph, intuitively it might seem obvious to perform traditional extension of ICA (abbreviated as TICA) by solving a similar system of linear equations corresponding to the adjacency matrix defined in equation (2.6) as follows,

$$\mathbf{L} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \mathbf{t} \tag{3.15}$$

where $\mathbf{L}$, the indicator vector $[\mathbf{x}\ \mathbf{y}\ \mathbf{z}]^T$ and $\mathbf{t}$ are similarly defined for the tripartite graph. However, by doing so the tripartite graph actually ends up being a bipartite graph of Y and $\{X\ \&\ Z\}$. This can be seen by shifting the X vertices on to the side of the Z vertices as illustrated in Fig.4. Due to this, we will be unable to distinguish between cutting a X-Y edge and an Y-Z edge and is thus a conceptual misrepresentation of the star structure as in Fig.3. An alternative approach is to use a weighting parameter α to prevent the mixing of X and Z vertices by defining the adjacency matrix of the tripartite graph as follows,

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{A}^T & \mathbf{0} & \alpha\mathbf{B} \\ \mathbf{0} & \alpha\mathbf{B}^T & \mathbf{0} \end{bmatrix} \tag{3.16}$$

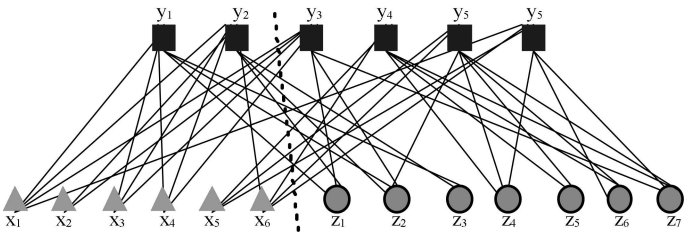

Figure 4. Traditional extension of ICA (TICA) for partitioning the tripartite graph ends up actually partitioning a bipartite graph of Y and $\{X \ \& \ Z\}$

However, as we demonstrate in Section 5.1, the numerical weightage is unable to prevent the problem. Moreover, even if it works on a particular dataset, it is not possible to decide the numerical weight *a priori* and will not work across other datasets.

To overcome the ill-partitioning of Fig.4, we propose the Consistent Isoperimetric High-order Co-clustering (CIHC) to partition the star-structured tripartite graph by considering it as two bipartite graphs coupled together. It is easy to see that applying ICA separately on the two bipartite graphs will result in two different partitioning results on Y. In order to achieve consistent results, we need to partition the two bipartite graphs simultaneously. That is, X-Y bipartite graph needs to be partitioned under the constraints enforced on Y by Z while, the partitioning of Y-Z bipartite graph has to be under the constraints enforced on Y by X. In other words, we achieve consistent partitioning of Y under the constraints that the partitioning of X-Y or Y-Z need not be optimal. By doing so, we are able to perform co-clustering of X, Y, Z under the constraints of the star-structure.

Applying ICA to the X-Y bipartite graph, we get,

$$\mathbf{L}^{(xy)} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \mathbf{t}^{(xy)} \tag{3.17}$$

Similarly, Y-Z bipartite graph yields us,

$$\mathbf{L}^{(yz)} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \mathbf{t}^{(yz)} \tag{3.18}$$

Next, we combine the above two system of linear equations as follows,

$$\begin{bmatrix} \mathbf{L}^{(xy)} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{(yz)} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{t}^{(xy)} \\ \mathbf{t}^{(yz)} \end{bmatrix} \tag{3.19}$$

$$\mathbf{F}\,\mathbf{r} = \mathbf{v}$$

$\mathbf{F}$ is not a square matrix, i.e. this is an overdetermined system of linear equations where the number of equations is more than the number of variables. Overdetermined system of linear equations is usually inconsistent and does not have any solution. However, many least squares methods exist to approximate the solution[26]. We adopted the QR decomposition method due to its simplicity and efficiency to solve equation

(3.19). Notice that, any other method can be employed as well. Amongst the common methods for cutting the indicator vector are the median cut and the ratio cut[27]. Median cut uses the median of the indicator vector $\mathbf{r}$ as the *splitting value* to produce equally sized partitions while ratio cut chooses one such that the resulting partitions have the lowest isoperimetric ratio of the graph indicating optimal partitioning. As our goal is not to necessarily produce equally sized clusters, we employ the ratio cut to get a bipartition.

In an abstract level, a star-structured k-partite graph can be considered as a generalized version of a tripartite graph. For simplicity, although we present the algorithm for a tripartite graph, its extension to a k-partite graph for high-order star-structured co-clustering is trivial.

3.1 Algorithm summary

The main steps of CIHC can be summarized as follows:

1. Using weight matrix $\mathbf{A}$ of the X-Y bipartite graph, construct the Laplacian matrix $\mathbf{L}^{(xy)}$ and vector $\mathbf{t}^{(xy)}$.
2. Similarly, using weight matrix $\mathbf{B}$ of the Y-Z bipartite graph, construct $\mathbf{L}^{(yz)}$ and $\mathbf{t}^{(yz)}$.
3. Using equation (3.19), construct $\mathbf{F}$, $\mathbf{r}$ and $\mathbf{v}$, and solve the overdetermined system of linear equations $\mathbf{F}\,\mathbf{r} = \mathbf{v}$.
4. Employ ratio cut on $\mathbf{r}$ to get the partitions.

3.2 Advantages of CIHC over CBGC

Both CIHC and CBGC partition the star-structured tripartite graph by considering it as a fusion of the two bipartite graphs. However, as we explain below the CIHC framework has a number of advantages over CBGC. In CBGC, the spectral objective functions of the two bipartite graphs are transformed into single-objective function of equation (2.7) by expressing it as a weighted linear combination. As argued in Ref.[28], this is a very ad-hoc approach and not a principled one. The two objective functions represent two different kinds of information, viz., the correlation between X-Y and Y-Z data types. Hence, instead of converting the original dual-objective problem into a single objective problem, one should use a dual-objective algorithm directly. Another drawback of CBGC, is that its performance is heavily dependent on 3 parameters, β, θ_1 and θ_2. β is the weighting parameter used in the linear combination of objective functions while θ_1 and θ_2 are parameters used to put constraints on the SDP bound controllers. The problem here is that these three parameters have to be predetermined. In their work, the authors test the performance of CBGC on a particular dataset by varying the values of the parameters and choose the best values[14]. As we show in our results (Section 5), this approach for tuning the CBGC parameters works only for that dataset and performs poorly on the other datasets. In the real world application for heterogeneous co-clustering, it is impossible to know what parameter values are to be chosen to achieve optimum results. On the other hand, CIHC is a completely parameter-less approach and does not require *a priori* specification of any parameters. In terms of computational complexity, CIHC is very

efficient compared to CBGC as it only requires a simple solution to a sparse system of linear equations.

4 Theoretical Analysis

4.1 Time complexity of CIHC

Computational time required by CIHC depends on the solution to equation (3.19). In particular, the time complexity is dependent on the number of non-zero entries in $\mathbf{L}^{(xy)}$ and $\mathbf{L}^{(yz)}$, which asymptotically is $O(|E|)$ where E is the set of edges in the tripartite graph. Note that, this only measures the time complexity to compute the indicator vector. We also need to include the time complexity to employ the ratio cut which is of the order of $O(h\,logh)$ where $h = |X| + |Y| + |Z|$. Factoring this in, the time complexity of CIHC is $O(|E| + h\,logh)$. Empirical results on computational speed are presented in Section 5.3.

4.2 Sensitivity analysis

In this section, we analyze the sensitivity of CIHC and CBGC with respect to a general parameter ρ.

4.2.1 CIHC

Recall from equation (3.19), CIHC requires a solution to a sparse system of linear equations represented by,

$$\mathbf{F}\ \mathbf{r} = \mathbf{v} \tag{4.20}$$

Differentiating this equation with respect to ρ,

$$\mathbf{F}\frac{\delta \mathbf{r}}{\delta \rho} = -\mathbf{r}\frac{\delta \mathbf{F}}{\delta \rho} + \frac{\delta \mathbf{v}}{\delta \rho} \tag{4.21}$$

For a given solution to equation (4.20), $\mathbf{F}$ and $\mathbf{r}$ are known and $\dfrac{\delta \mathbf{F}}{\delta \rho}$ can be determined analytically. In order to determine the derivative at point $\mathbf{r}$, $\dfrac{\delta \mathbf{r}}{\delta \rho}$ can be solved for as a system of linear equations.

4.2.2 CBGC

The CBGC objective function is,

$$\min\left\{\beta\frac{\mathbf{q}^T\mathbf{L}^{(xy)}\mathbf{q}}{\mathbf{q}^T\mathbf{D}^{(xy)}\mathbf{q}} + (1-\beta)\frac{\mathbf{p}^T\mathbf{L}^{(yz)}\mathbf{p}}{\mathbf{p}^T\mathbf{D}^{(yz)}\mathbf{p}}\right\} \text{ subject to certain constraints} \tag{4.22}$$

If we drop the constant weighting parameter β and consider only the first term, we get,

$$\min\left\{\frac{\mathbf{q}^T\mathbf{L}^{(xy)}\mathbf{q}}{\mathbf{q}^T\mathbf{D}^{(xy)}\mathbf{q}}\right\} \tag{4.23}$$

From spectral graph partitioning[18,19], we know that the solution for minimizing this term is the eigenvector corresponding to second smallest eigenvalue λ_2 of the generalized eigenvalue problem,

$$\mathbf{L}^{(xy)}\ \mathbf{q} = \lambda_2\ \mathbf{D}^{(xy)}\ \mathbf{q} \tag{4.24}$$

Differentiating this equation with respect to ρ,

$$\mathbf{q}\frac{\delta \mathbf{L}^{(xy)}}{\delta \rho} + \mathbf{L}^{(xy)}\frac{\delta \mathbf{q}}{\delta \rho} = \mathbf{D}^{(xy)}\ \mathbf{q}\frac{\delta \lambda_2}{\delta \rho} + \lambda_2 \mathbf{q}\frac{\delta \mathbf{D}^{(xy)}}{\delta \rho} + \lambda_2 \mathbf{D}^{(xy)}\frac{\delta \mathbf{q}}{\delta \rho} \tag{4.25}$$

Using Rayleigh quotient and the Chain rule, it is possible to calculate $\frac{\delta \lambda_2}{\delta \rho}$.

The Rayleigh quotient is,

$$\lambda = \frac{\mathbf{q}^T \mathbf{L}^{(xy)} \mathbf{q}}{\mathbf{q}^T \mathbf{q}} \tag{4.26}$$

Applying Chain rule,

$$\frac{\delta \lambda_2}{\delta \rho} = \frac{\delta \lambda_2}{\delta \mathbf{q}}\frac{\delta \mathbf{q}}{\delta \rho} \tag{4.27}$$

In the above equation, $\frac{\delta \lambda_2}{\delta \mathbf{q}}$ can be calculated from equation (4.26) as,

$$\frac{\delta \lambda_2}{\delta \mathbf{q}} = 2\mathbf{L}^{(xy)}\mathbf{q}(\mathbf{q}^T\mathbf{q})^{-1} - 2\mathbf{q}^T\mathbf{L}^{(xy)}\mathbf{q}(\mathbf{q}^T\mathbf{q})^{-2}\mathbf{q} \tag{4.28}$$

From equation (4.25), since all the terms are either known or can be calculated analytically, we get a system of linear equations which may be solved for $\frac{\delta \mathbf{q}}{\delta \rho}$.

If we now consider the second term in equation (4.22), and proceed similarly, we get

$$\mathbf{p}\frac{\delta \mathbf{L}^{(yz)}}{\delta \rho} + \mathbf{L}^{(yz)}\frac{\delta \mathbf{p}}{\delta \rho} = \mathbf{D}^{(yz)}\ \mathbf{p}\frac{\delta \lambda_2}{\delta \rho} + \lambda_2 \mathbf{p}\frac{\delta \mathbf{D}^{(yz)}}{\delta \rho} + \lambda_2 \mathbf{D}^{(yz)}\frac{\delta \mathbf{p}}{\delta \rho} \tag{4.29}$$

We can analyze the effect of a specific parameter, e.g., edge weight, by substituting for the general parameter ρ. Equations (4.21), (4.25) and (4.29) show that the derivative of the CIHC solution is never degenerate. On the other hand, the CBGC solution may be degenerate depending on the value of λ_2 and the state of its corresponding eigenvector.

5 Experiments and Results

5.1 Toy dataset

In Section 3, we discussed the need for partitioning the star-structured tripartite graph by simultaneously partitioning the two bipartite graphs. We now illustrate this on a toy dataset that TICA does not yield optimum results. For this, we created a tripartite graph shown in Fig.5 having 3 X, 8 Y and 6 Z vertices with uniform weights along all the edges. It is easy to infer the ideal partitioning of this graph, shown in the Figure using a dotted line. We partitioned this graph using TICA with adjacency matrix defined in equation (3.16) and CIHC. Since this graph had uniform weights, unit volume as defined in equation (3.10) was employed for this experiment. In Fig.6, we show the results achieved by TICA when we varied the value of α from 0.01 to 1000. The X-axis has the vertices in the order of X, Y and Z with the dotted line separating each of the data types. Embedding value for each vertex in the indicator vector is plotted along the Y-axis. The two pattern plots ($*$ and $\triangle$) represent the two clusters obtained. These results clearly show that in spite of increasing edge

weights for one of the bipartite graphs drastically over the other, TICA is still unable to distinguish between cutting a X-Y edge from a Y-Z edge on a simple graph. On the other hand, Fig.7 shows the results achieved using CIHC. We can see that, CIHC achieves perfect clustering for all the vertices. Through this toy data experiment, we have shown the need to consistently apply isoperimetric co-clustering simultaneously to the two bipartite graphs.

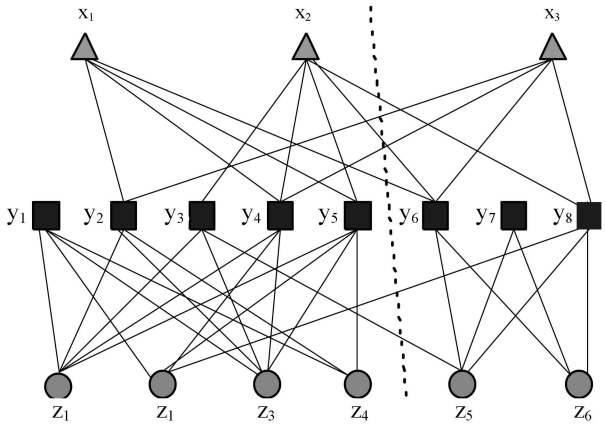

Figure 5. Toy dataset consisting of 3, 8 and 6 vertices of X, Y and Z, respectively with uniform weights along all edges. The dotted line shows the ideal cut for partitioning this graph

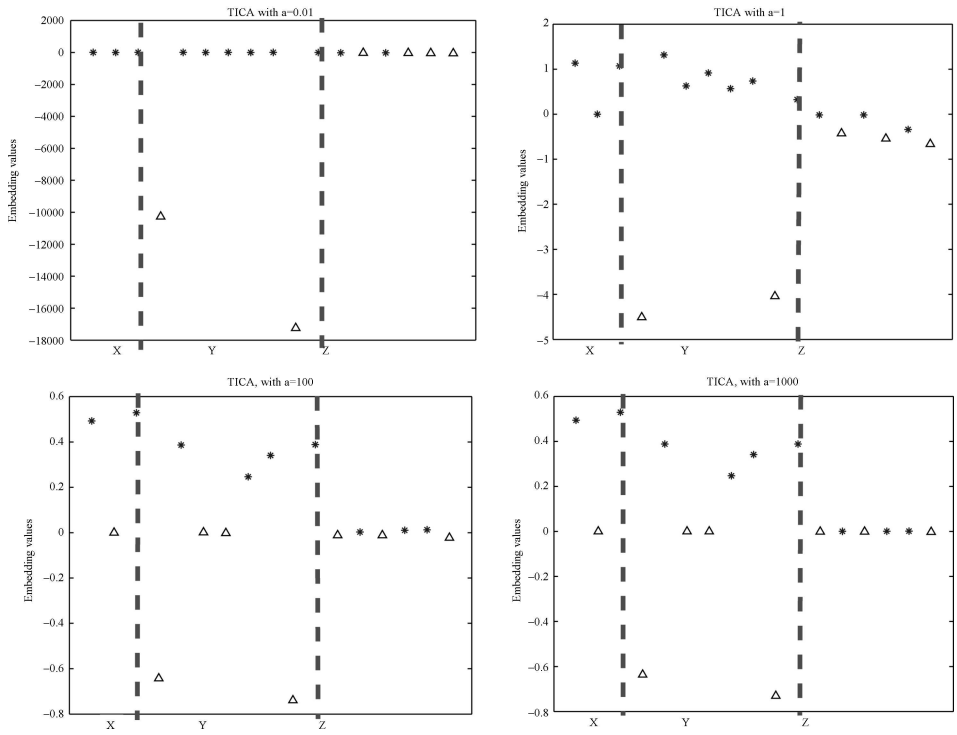

Figure 6. TICA results for partitioning the toy dataset. Each sub-figure shows the results for $\alpha = 0.01$, 1, 100 and 1000, respectively

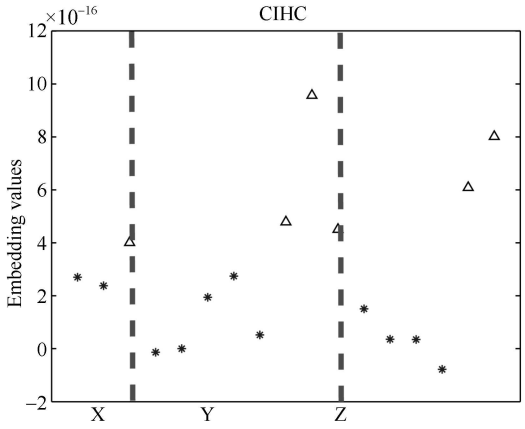

Figure 7. CIHC results for partitioning the toy dataset.
We are able to get perfect partitioning for each of X, Y and Z vertices

5.2 Real dataset

The relationship between categories, documents and words is star-structured and has been previously used for hierarchical taxonomy classification for text categorization[29]. For our experiments, we have utilized the dataset used in[30]*. In particular, we have made use of the oh10 & oh15 datasets that are from the OHSUMED collection, re0 & re1 datasets that are from the Reuters- 21578 text categorization test collection Distribution 1.0 and the data set *wap* from the WebACE project.

As mentioned in Section 3.2, CBGC is a very parameter-dependent framework which relies heavily on the values of β, θ_1 and θ_2. To decide on the values for these 3 parameters, we followed the approach adopted by the authors in Ref.[14]. We constructed a small dataset by selecting *People* and *Television* categories from the *wap* dataset. We performed selection of the words according to[31] to get 862 words. After this, some documents having sparse representation were removed leaving behind 298 documents. Of these, 168 documents were from the *People* category while the remaining 130 belonged to the category *Television*. We ran CBGC algorithm on this dataset by varying values of the 3 parameters between 0 and 1. Setting the value of β to either extreme caused ill-partitioning for one of the bipartite graphs. Consequently, we fixed β to a value of 0.5 so as to give equal importance to both the bipartite graphs. For θ_1 and θ_2, we observed that performance of CBGC improves considerably as the values of these 2 parameters approach 1, particularly from 0.8 onwards. On this dataset, we achieved best results by setting them to 1. The CBGC results with $\beta = 0.5$ and $\theta_1 = \theta_2 = 1$ are shown in Fig.8. The 3 sub-figures show the partitioning obtained for categories, documents and words, respectively. The dotted line in the second sub-figure separates the documents from the two categories. Good document clustering should partition documents on either side of the line into different clusters. We can see that for these parameters, the algorithm has been able to achieve good

*http://www.cs.umn.edu/~han/data/tmdata.tar.gz

clustering results. However, as argued by us previously, setting parameter values on one dataset is a very dataset specific approach as these values do not work on other datasets. We demonstrate this on two datasets shown in Table 1.

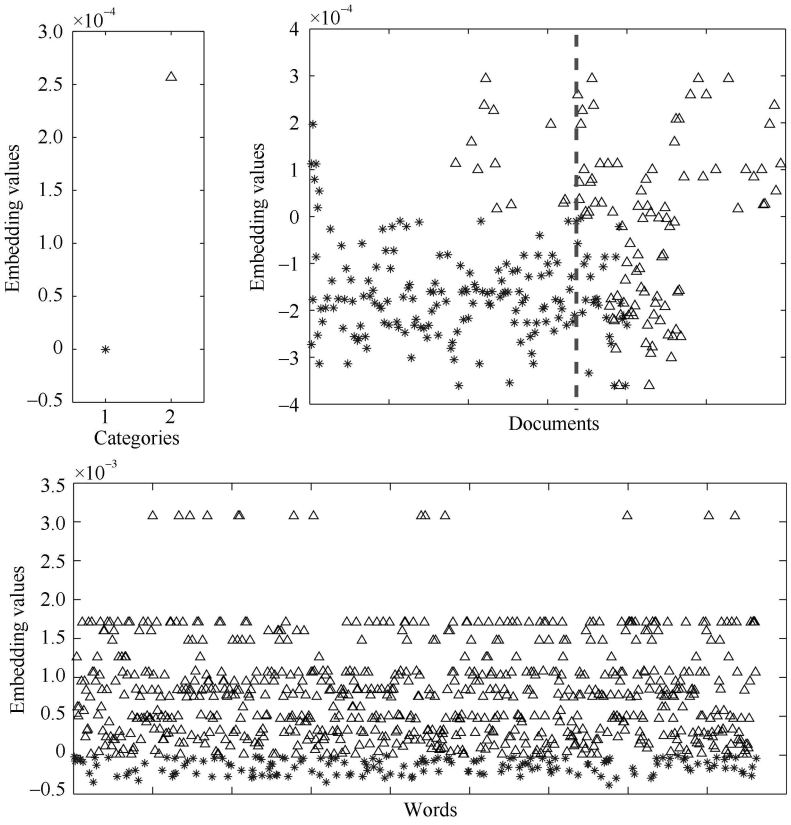

Figure 8. CBGC results on People-Television dataset

Table 1 Summary of the datasets

Dataset Notation	Dataset	Categories	Category Notation
DS1	oh15	Blood Coagulation Factors	C_{11}
		Blood Vessels	C_{12}
	re1	Coffee	C_{13}
		Sugar	C_{14}
DS2	oh10	Eating	C_{21}
		Nutrition	C_{22}
	re0	Money	C_{23}
		Retail	C_{24}
		Trade	C_{25}

DS_1 dataset consisted of 4 categories - 2 each from $oh15$ and the $re1$ datasets. We performed similar preprocessing to retain 527 words and 292 documents. It is easy to infer that the expected clustering on the categories is $\{C_{11}, C_{12}\}$ and $\{C_{13}, C_{14}\}$. In Fig.9, we report the results of CBGC on this dataset with the values of

the 3 parameters set as before. The algorithm is able to get perfect clustering on the categories. In the documents clustering sub-figure, the left and right of the dotted line separates the documents from the first two and next two categories, respectively. As can be seen, while the algorithm performs well for clustering categories, it has not been able to achieve optimum clustering of the documents with the set parameter values. To evaluate the partitioning of the words, we calculated the document frequency (note: document frequency is the number of documents in which a word appears) of the top 20 words in both the clusters. While the CBGC results shows quite a few words with tall bars, a number of words have a document frequency of zero. This means, that these words do not belong to any of the documents in the cluster and have been misclassified. On the other hand in Fig.10, CIHC completely outperforms CBGC on the same dataset. The algorithm has achieved optimum clustering for each of categories, documents and words.

We now compare the results of the two algorithms on the DS_2 dataset consisting of 5 categories - 2 from $oh10$ and 3 from $re0$ dataset. After preprocessing, the dataset consisted of 549 words and 507 documents. In this case, the expected clustering on the categories is $\{C_{21}, C_{22}\}$ and $\{C_{23}, C_{24}, C_{25}\}$. Figure 11 shows the results we obtained by employing CBGC. The algorithm performs very poorly in clustering the categories

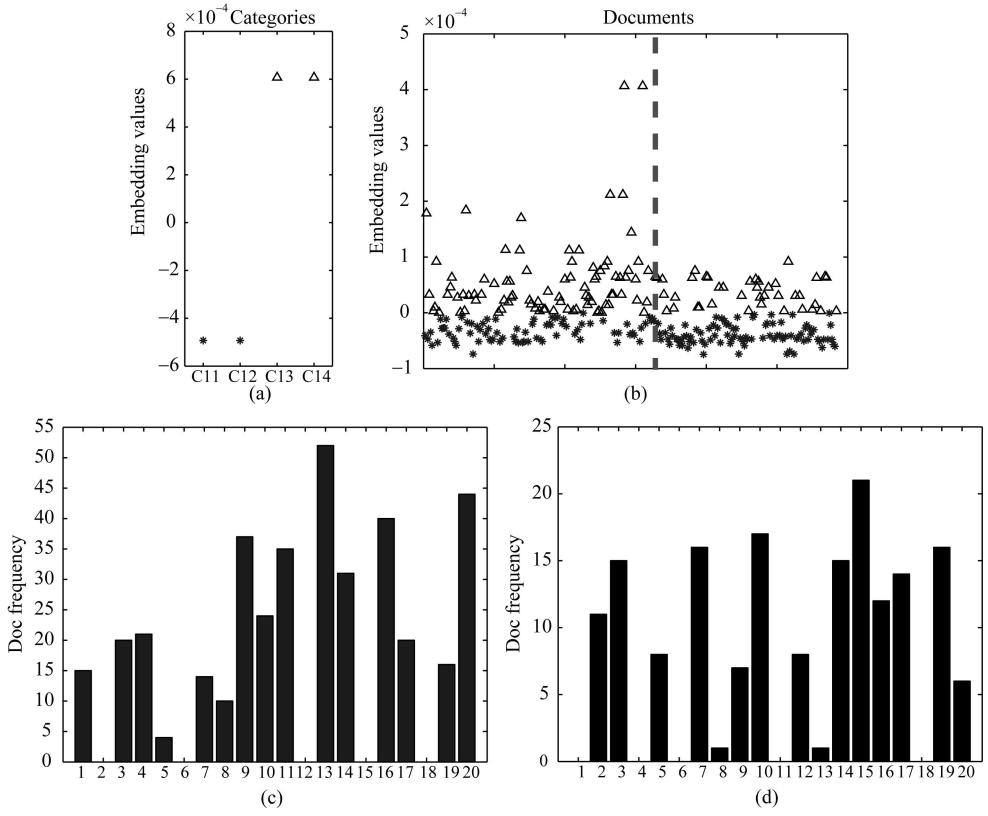

Figure 9. CBGC results on dataset DS_1. Clustering of (a)categories and (b)documents.

Doc frequency for top 20 words in (c)Cluster $*$ and (d)Cluster $\triangle$

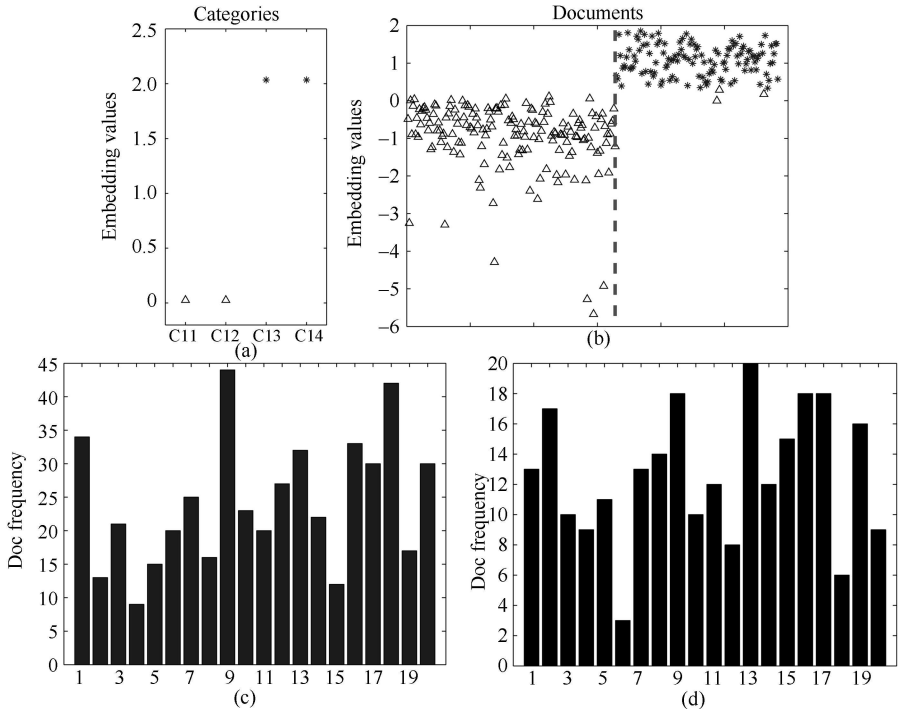

Figure 10. CIHC results on dataset DS_1. Clustering of (a)categories and (b)documents. Doc frequency for top 20 words in (c)Cluster $*$ and (d)Cluster $\triangle$

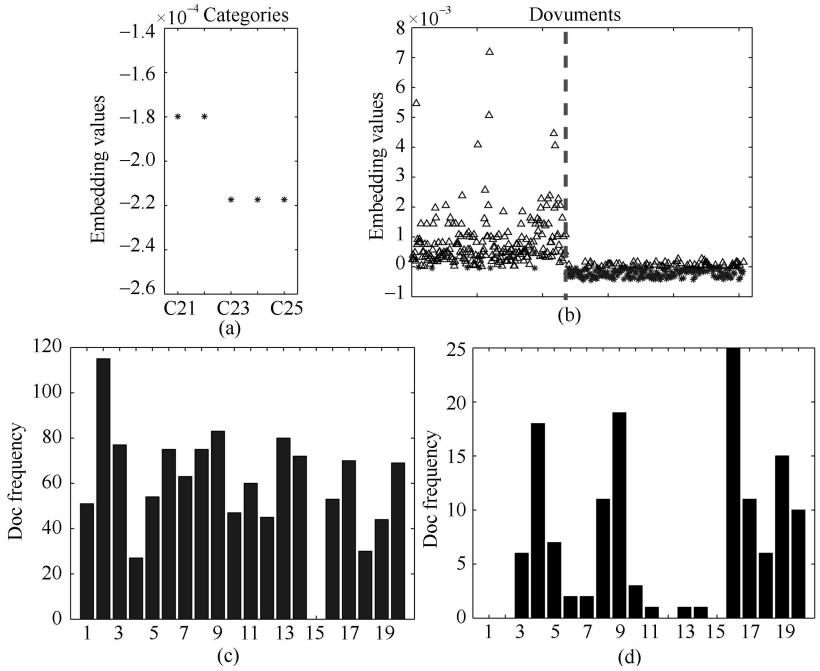

Figure 11. CBGC results on dataset DS_2. Clustering of (a)categories and (b)documents. Doc frequency for top 20 words in (c)Cluster $*$ and (d)Cluster $\triangle$

and has assigned all to one cluster. For documents clustering, although the algorithm does misclassify a few documents, it is still able to achieve quite satisfactory results. The words clustering results are biased to one cluster with many words classified to it. This can be seen from good clustering of words in one cluster and very poor in the other. The performance of CIHC is shown in Fig.12. The algorithm correctly partitions the categories and has achieved perfect clustering of the documents. Also, for words clustering, none of the top 20 words have been misclassified.

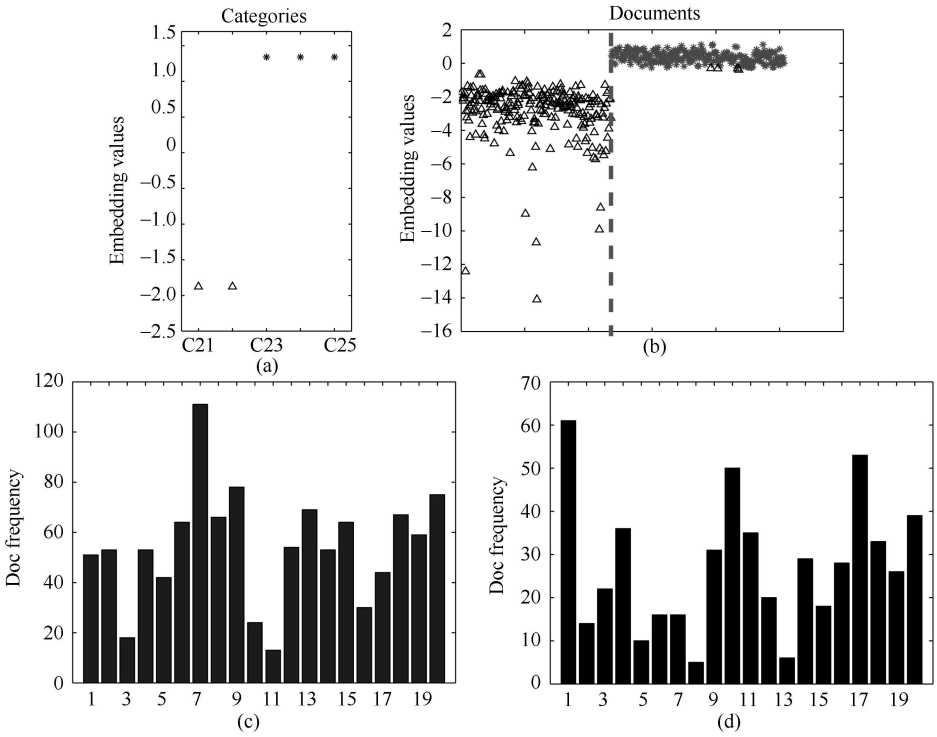

Figure 12. CIHC results on dataset DS_2. Clustering of (a)categories and (b)documents. Doc frequency for top 20 words in (c)Cluster $*$ and (d)Cluster $\triangle$

This shows that the CBGC approach of tuning the parameters on one dataset as performed in Refs.[13,14] do not work for other datasets. Moreover, in real world scenario, it is unreasonable to expect the parameter values to be predetermined for clustering as the ground truth is not known. On both the datasets, CIHC was able to convincingly outperform CBGC.

5.2.1 Performance in the presence of noise

We now compare the stability of CIHC and CBGC by evaluating the performance in the presence of Gaussian additive and multiplicative noise. Additive noise had zero mean while multiplicative had a mean of 1. The goal of graph partitioning algorithms in general is to obtain partitions with a low isoperimetric ratio. In other words, lower isoperimetric ratios demonstrates optimal partitioning. In 13, we plot the isoperimetric ratios of partitioning the 2 datasets using CIHC and CBGC as the variance of the noise was increased from 1 to the maximum value in the data matrices. The first two figures are for the additive noise on the two datasets and the

next two are for multiplicative noise. We see that inspite of the varying amounts and kinds of noise in the data, CIHC is able to perform optimal partitioning indicated by its low isoperimetric ratio. Second noticeable fact is in regards to stability. Rising ratios as the variance increases indicates that the performance of the algorithm is decreasing. However, fluctuating ratios indicates instability and inconsistency to partition optimally. While the ratios of CIHC fluctuate in limits, CBGC ratio fluctuation clearly shows instability. To demonstrate this, we calculated the standard deviation of the isoperimetric ratios of both the algorithms for partitioning in the presence of the noise. These results are shown in Table 2. Higher standard deviation of CBGC indicates instability to partition the noisy datasets.

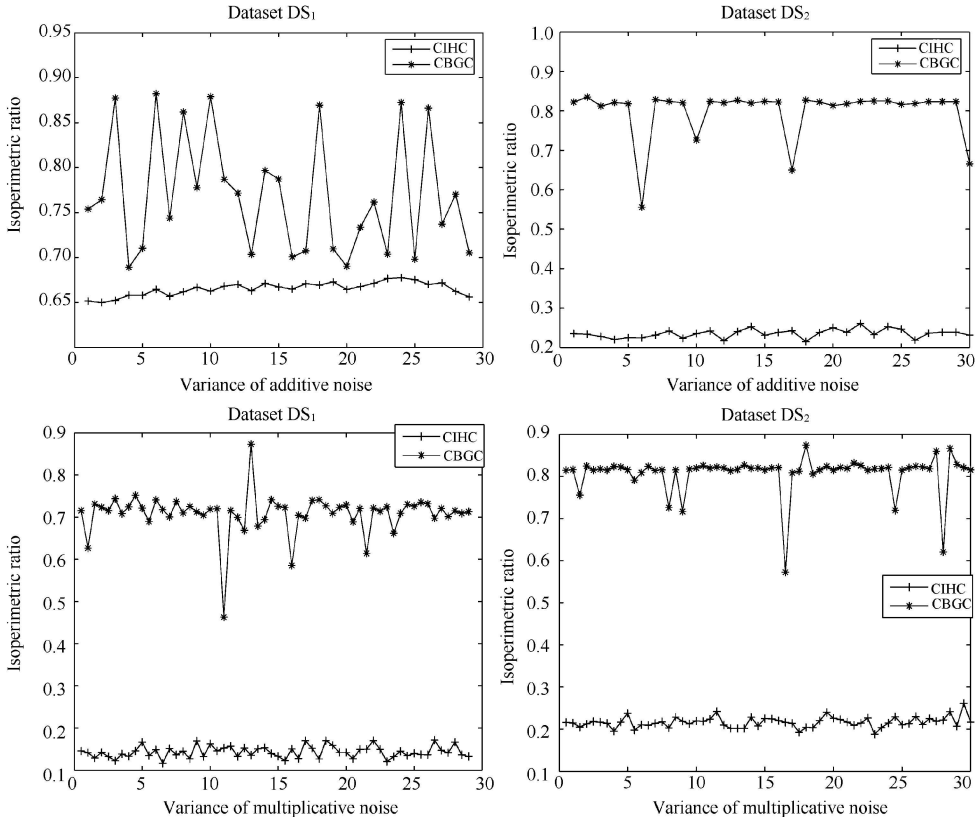

Figure 13. Performance of CIHC andCBGC in the presence of noise. First two figures are for the additive noise on the two datasets and the next two are for multiplicative noise.

Table 2 Standard deviation of the Isoperimetric Ratios of CIHC and CBGC over the noisy datasets

Partitioning with	Data	CIHC	CBGC
Additive noise	DS1	0.0074	0.0671
	DS2	0.0110	0.0641
Multiplicative noise	DS1	0.0137	0.0493
	DS2	0.0128	0.0477

5.3 Computational speed comparison

We now compare the computational speed of CIHC with CBGC. The time take by both algorithms is dependent on the sparseness of the two data matrices. In other words, it takes more time to partition a densely connected tripartite graph compared to a sparsely connected one. For this reason, we considered the worst case scenario of a fully connected tripartite graph (with uniform weights) where every vertex in both the bipartite graphs is connected with all other vertices of the other type. Since the time required to cut the indicator vector is the same for both algorithms, we compare on the basis of the time required to calculate the indicator vector. The algorithms were implemented using MATLAB. For CBGC implementation, we made use of the SDP library SDPA-M*†. The experiment was performed on a machine with a 3 GHz Intel Pentium 4 processor with 1 GB RAM. In Fig.14, we plot the time required by the algorithms as the number of vertices in the fully connected tripartite graph increases. Time for CIHC gradually increases with the number of vertices. For the maximum number of vertices we increased to - about almost $4,000$, CIHC required about 98 seconds only. CBGC on the other hand, was unable to keep up with CIHC. As can be seen, the time required by CBGC really shoots up for a few hundred vertices in the graph. Moreover, CBGC is unscalable and is unable to handle larger sized graphs, which was also verified by Refs.[11,12]. In our experiment, CBGC was unable to handle graphs with more than $1,500$ vertices. This clearly demonstrates the computational efficiency of CIHC and the potential for applicability in large-scale real-world applications.

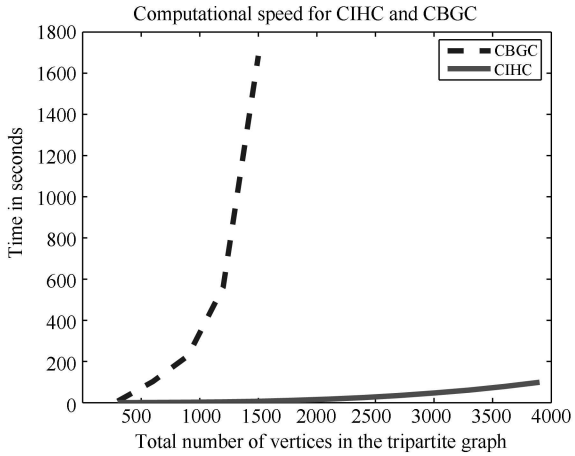

Figure 14. Computational speed comparison of CIHC with CBGC. The time required by each of the algorithms to compute the indicator vector are displayed for increasing number of vertices in the tripartite graph

6 Conclusions and Future Work

In this paper, we addressed the star-structured co-clustering problem which is a specific case of heterogeneous co-clustering. In co-clustering problems of these type, a central data type is connected to all other data types. We proposed the

*http://grid.r.dendai.ac.jp/sdpa

CIHC framework that simultaneously co-clusters all the data types by partitioning a star-structured k-partite graph. Our algorithm is very quick as it requires a simple solution to a sparse system of overdetermined linear equations. Theoretical analysis and extensive experiments performed on toy and real datasets verify the effectiveness of the proposed framework.

In future work, there are a number of directions we are actively pursuing. Currently, in order to get more than two partitions, we recursively apply CIHC which is a common approach in many other graph partitioning algorithms[7,8,14,16]. We are currently investigating methods to get more than two clusters directly. We also plan to work on general heterogeneous co-clustering without the constraints of the star-structure.

References

[1] Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys, 1999, 31(3): 264–323, 1999.
[2] Dhillon IS, Mallela S, Modha DS. Information-theoretic co-clustering. In: Proc. of KDD. 2003.
[3] Cai R, Lu L, Hanjalic A. Unsupervised content discovery in composite audio. In: ACM Multimedia. 2005.
[4] Banerjee A, Dhillon IS, Ghosh J, Merugu S, Modha DS. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In: Proc. of KDD. 2004.
[5] George T, Merugu S. A scalable collaborative filtering framework based on co-clustering. In: Proc. of ICDM. 2005.
[6] Dhillon IS. Co-clustering documents and words using bipartite spectral graph partitioning. In: Proc. of KDD. 2001.
[7] Zha H, He X, Ding CHQ, Simon H, Gu M. Bipartite graph partitioning and data clustering. In: Proc. of CIKM. 2002.
[8] Rege M, Dong M, Fotouhi F. Co-clustering documents and words using bipartite isoperimetric graph partitioning. In: Proc. of ICDM. 2006.
[9] Zeng H, Chen Z, Ma W. A unified framework for clustering heterogeneous web objects. In: Proc. of WISE. 2002.
[10] Wang J, Zeng H, Chen Z, Lu H, Tao L, Ma W. Recom: reinforcement clustering of multi-type interrelated data objects. In: Proc. of SIGIR. 2003.
[11] Long B, Zhang Z, Wu X, Yu PS. Spectral clustering for multi-type relational data, in Proc. of ICML, 2006.
[12] Long B, Wu X, Zhang Z, Yu PS. Unsupervised learning on k-partite graphs. In: Proc. of KDD. 2006.
[13] Gao B, Liu TY, Zheng X, Cheng QS, Ma WY. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering. In: Proc. of KDD. 2005.
[14] Gao B, Liu TY, Qin T, Zheng X, Cheng QS, Ma WY. Web image clustering by consistent utilization of visual features and surrounding texts. In: ACM Multimedia. 2005.
[15] Boyd S, Vandenberghe L. Convex Optimization. Cambridge University Press, 2004.
[16] Grady L, Schwartz EL. Isoperimetric graph partitioning for image segmentation. IEEE Trans. on PAMI, 2006, 28(3): 469-475.
[17] Grady L, Schwartz EL. Isoperimetric partitioning: A new algorithm for graph partitioning. SIAM Journal on Scientific Computing, 2006, 27(6): 1844-1866.
[18] Chung FRK. Spectral Graph Theory, American Mathematical Society, 1997.
[19] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans. on PAMI, 2000, 22(8): 888–905.
[20] Golub GH, Van-Loan CF. Matrix Computations. John Hopkins Press, 1989.
[21] Kumar R, Mahadevan U, Sivakumar D. A graph-theoretic approach to extract storylines from search results. In: Proc. of KDD. 2004.
[22] Ding CHQ. Unsupervised feature selection via two-way ordering in gene expression analysis.

Bioinformatics, 2003, 19: 1259–1266.

[23] Dodziuk J. Difference equations, isoperimetric inequality and the transience of certain random walks. Trans. of the American Mathematical Society, 1984, 284: 787–794.

[24] Dodziuk J, Kendall WS. Combinatorial laplacians and isoperimetric inequality, From Local Times to Global Geometry, Control and Physics of Pitman Res. Notes Math. Ser., Longman Sci. and Tech., 1986, 150: 68–74.

[25] Mohar B. Isoperimetric numbers of graphs. Journal of Combinatorial Theory, Series B, 1989, 47: 274–291.

[26] Lawson CL, Hanson RJ. Solving Least Squares Problems. Soc for Industrial and Applied Math, 1995.

[27] Hagen L, Kahng AB. New spectral methods for ratio cut partitioning and clustering, IEEE Trans. on CAD of Integrated Circ. & Sys., 1992, 11(9): 1074–1085.

[28] Freitas AA. A critical review of multi-objective optimization in data mining: A position paper. SIGKDD Explor. Newsl., 2004, 6(2): 77–86.

[29] Gao B, Liu TY, Feng G, Qin T, Cheng QS, Ma WY. Hierarchical taxonomy preparation for text categorization using consistent bipartite spectral graph copartitioning. IEEE Trans. on KDE, 2005, 17(9): 1263–1273.

[30] Han EH, Karypis G. Centroid-Based document classification: Analysis & experimental results. In: Proc. of PKDD. 2000.

[31] Yang Y, Pedersen JO. A comparative study on feature selection in text categorization. In: Proc. of ICML. 1997.